Digital Signal Processor

FIELD OF THE INVENTION

The present invention relates to a digital signal processor that is driven by events. The processor comprises a central arithmetical unit, a register, a controller, an instruction memory, and input/output devices.

DESCRIPTION OF PRIOR ART

Depending on the architecture of a digital signal processor, the same type of functionality can be performed quite differently. In control driven computing the basis is the predefined control procedure, stored as a program in machine code, describing in what sequence the information shall be processed. In a data driven architecture the processing unit in the computer operates only if all input data is present in an input buffer and enough space is available in the output buffer to store the result. In a demand driven architecture, execution is initiated by demand of output data. In an event driven architecture event and time is introduced. Each unit start processing immediately when input data arrive, recognized as detection of an input event. Events are conventionally handled in the operative system. Event driven processing is primarily used in reconfigurable systems with lower performance. The lower performance is a trade-off to the possibility to alter the behavior in the system by means of programming.

Compilers of today cannot map time or event expressed in software to lower level than the real-time operative system because time and event do not exist at the level below. The final execution of a single time or event expressed in the application software always ends up in the execution of a large number of machine code operations in sequence. This happens irrespective of computing architecture chosen for the application (control-, data-, demand- or event-driven). Consequently time performance for the final application is hard to guarantee because it is dependent of all simultaneous activities running on the processor and the real-time system. Achieved real-time performance is in general satisfying for mainstream applications, however it is too low real-time performance for more demanding applications. The consequence is that demanding applications cannot be implemented as a programmable application and executed on a digital signal processor.

Various real-time operative systems have been developed for the purpose of handling time and parallel processes. These systems provide a possibility of expressing time by means of

a real-time operative system call to a real-time clock. The real-time operative system also allows event to effect the program by means of routines for interrupt. Data-, demand-, and event-driven programmable behavioral is always possible to use to implement the application software on top of a platform consisting of a real-time operative system and a traditional control driven computing architecture. However, the event driven processing by means of the operative system cannot be performed with sufficient time resolution for many high-speed applications, e.g., radar. For these types of applications, event-driven processing is particularly favorable since the function of these type of applications many times depend upon when a specific action is performed. This can easily be expressed in an event-driven architecture.

SUMMARY OF THE INVENTION

It is an object of the present invention to solve the problems in conventional event driven digital signal processors. This object is achieved through the digital signal processor in accordance with claim 1.

The event driven signal processor includes an arithmetic/logical unit , an internal register, a controller, an instruction memory and input/output devices as any conventional signal processor. The instruction memory is arranged to include time performance constraints and events.

The controller in the event driven signal processor in accordance with claim 1 is arranged to suspend further processing of time performance constraints after initiating operations in an event control unit connected to the processor. The event control unit is arranged to recognize an event and to control the processing to be carried out as a consequence of the event. All processing is carried out under the time performance constraints. The controller resumes processing when advised by the event control unit.

The invention also includes the specific embodiments disclosed in claims 2-10. In a preferred embodiment, two or more event control units are arranged in the processor. This allows the controller to proceed with processing of time performance constraints in one event control unit after initiating operations in another event control unit.

The event driven signal processor in accordance with the invention provides for a reconfigurable event driven signal processor, which handle time constraints in an elegant way. The performance of the processor may easily be predicted deterministically without any uncertainty caused by cache misses or interfering real-time operative system. The event driven processor is particularly suitable for high-speed applications, e.g., radar applications, routers or network processing.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 discloses an architecture in a control driven processor extended to an event driven processing architecture in accordance with the invention

Fig. 2 discloses the content of a one pulse package instruction

Fig. 3 discloses the internal structure of an event control unit

Fig. 4 discloses a block diagram of a second embodiment of an event driven processor

Fig. 5 discloses a structure to implement high-resolution time delay

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

The preferred embodiments of the present invention will be described hereinafter with reference to the accompanying drawings.

Figure 1 discloses architecture for a first embodiment of an event driven processor 1 in accordance with the invention. The processor 1 is under control of a host processor, which is not disclosed in the figure. Input from the host processor is stored in the instruction memory 2. The processor 1 and the environment 8 form a real-time system. The instruction memory 2 holds operation code comprising logical operations as well as time constraints and events that will initiate operations within the processor 1. A signal memory 7, which can be a vector memory or any other suitable type of memory, stores signal data to be handled under the time-constraints in the processor 1. A controller 3 organizes reading and execution of the operation code in the instruction memory 2 so that everything is carried out in the correct sequence. Intermediate results are stored in a register 4 in conventional manner. An arithmetic and logical unit 5 is arranged to operate in a conventional manner. The processor 1 also includes an event control unit 6, which is capable of recognizing an event and to control the processing to be carried out as a consequence of the event while fulfilling the time performance constraints.

The controller 3 initiates operations in the event control unit 6. After initiating operation, the controller 3 suspends further processing of time performance constraints and awaits an alert from the event control unit 6. The event control unit 6 controls the vector memory 7 and defines access to the vector memory 7 so that data is extracted or stored at the right moment in time. When advised by the event control unit 6 following detection of an event, the controller 3 resumes processing by initiating a new operation in the event control unit 6. The event-driven behavior is expressed in the instruction memory 2 as a chain or sequence of events with associated actions. An action is defined as an achievement expected as a direct consequence of the event. Events are executed in direct sequence where a succeeding action is processed directly after a previous action is completed. Each event control unit 6 start processing immediately when input data arrive, recognized as detection of an input event. The computed result is presented at the output at a time defined by a corresponding time constraint. Input and output buffering is not required. The controller 3 is responsible for defining, through the event control unit 6, which part of the vector memory 7 that should be used for certain data. The event control unit 6 regulates when data is extracted or stored.

With the event driven processor 1 disclosed in fig 1, a vector memory 7 and an event control unit 6 allow events and time to be handled in the processor 1. With time constraints introduced in the processor 1, time critical functionality can be moved from the level above real-time operating system to the event control unit 6.

Figure 2 discloses an example of a typical event. The action is phase modulation illustrated by the square wave affecting the extracted data. At machine instruction level event and time can be expressed as a pulse package operation with four operands: *event, delay, vector* and *action.*

Event is an cperand that defines the event that initiate execution.

The *delay* operand defines a time constraint – a time interval to elapse between event and access to the vector memory 7 to extract or store data.

Vector is an operand that defines location in the vector memory 7.

The action operand defines future processing to be carried out on data after extraction or before storing in the vector memory 7.

An event control unit 6 is disclosed in figure 3. The event control unit 6 includes a package controller 9, which is a combined controller and time counter; an interface to the vector memory 7 and to external devices; and registers 10a, 10b to hold operands for pulse packages to facilitate momentary switch. An active register 10a holds the pulse package that is in process or to be processed at the next event. A buffer register 10b holds the subsequent pulse package to facilitate momentary switch between two subsequent pulse packages. Each register 10a, 10b is subdivided into four fields, each holding one operand. The *event* operand in the registers 10a, 10b selects which external or internal event signal that should initiate the execution. The delay field contains the time constraint. The *delay* operand is used to define a stop condition for the counter in the package controller 9. The *vector* operand is used by the package controller 9 to define the start and end position for signal data in the vector memory 7. The *action* field interfaces with external devices to control activity in these devices. A signal from the package controller 9 controls transfer of new cperands from the buffer register 10b to the active register 10a. Immediately after completion of the transfer, the package controller 9 will request new operands from the overlaying controller 3 using handshake signals. Execution of machine instructions in controller 3 is halted or resumed depending on the handshake signal.

Figure 4 shows a second embodiment of the architecture for an event driven processor 1. The processor 1 is placed between Bus #1, connecting to host processor, and Bus #2, passing parameters to event control units 6a, 6b and to external devices such as a modulator 11 included in the output data path. The instruction memory 2 has two logically separated parts 2a, 2b to enable update of the operation codes in the instruction memory 2 under processing. The controller 3 manages the execution of operation codes for the event control units 6a, 6b as stored in the instruction memory parts 2a, 2b as separate jobs. The vector memory 7 can be a four-port SRAM, where two ports are used for connection to data paths to external devices with simultaneous read and write. The other two ports are connected to Bus #1 for parallel signal analysis. The controller 3 manages the execution of operation codes needed for the two event control units 6a, 6b, one for receiver function and one for transmitter function. Operation codes for the two functions are stored as separate jobs in the instruction memory 2.

The event driven signal processor 1 is initialized by its host processor through Bus #1. The instruction memory 2 is loaded with code describing what actions to take when events

arrive. The host processor starts an initial sequence of code to set up other hardware. The interconnection of the host processor and the event driven processor 1 by means of the instruction memory 2 enables a programmable behavior in the event driven processor 1. During updating of the instruction memory 2, updating is carried out within a first part 2a of the instruction memory 2, which is not involved in the execution of operation code. When the update has been terminated for the first part 2a, the execution of operation code is carried out from this first part 2a. Updating of the second part 2b is then initiated. After programming, a first initiation sequence is executed. After initiation, the event driven processor 1 suspends its processing and waits for an event to occur. When an event is detected, the pulse package operation stored in the active register 10a is executed. After execution of the pulse package is completed, the involved event control unit 6a/6b immediately initiates execution of the pulse package stored in the buffer register 10b. A request is sent to the controller 3 for new pulse packages. The request is accompanied by information of a job number, which is used by the controller 3 to resume execution of the corresponding job code in the instruction memory 2. When a job is processed, each instruction tells the controller 3 if it should continue, wait and then continue or wait and then restart. More than one event can be handled in parallel. Actions taken upon events are reconfigurable by updating the code in the instruction code area. A first part 2a is being executed while the second part 2b is updated from Bus #1. It is possible to update behavior for one job or more by using the logical separation in a first and second part 2a, 2b in the instruction memory 2. For example, while the first part 2a is updated from host processor, the second part 2b is used for event actions. Further signal processing of data from the vector memory 7 is controlled via bus #2 to the modulator 11.

High programmable time delay resolution is obtained by splitting the implementation of the delay field 10a2 in the active register 10a into two parts: a most significant part and a least significant part. The most significant part is passed to the package controller 9. A counter in the package controller 9 produces a delay as defined by the most significant part. Following the delay, data is extracted from the vector memory 7 and stored in an output buffer 13, which is part of the vector memory 7. The least significant part is passed to the high-speed controller 12 during the data extraction from the vector memory 7. A counter in the high-speed controller 12 produces a delay in a signal, which is passed from the package controller 9 to the high-speed controller 12 during data extraction from the vector memory 7. The high-speed controller 12 operates at the clock frequency and the

counter in the package controller 9 operates at a lower frequency corresponding to the number of bits in the least significant part. Output data is stored in the buffer 13 until a signal: load_out_buffer, the delayed signal from the high-speed controller, becomes active. This signal is delayed as defined by the least significant part. Write is handled in a similar way.

The event driven processor 1 in accordance with the invention allows large data packages transmitted with high speed, e.g., radar signals, to be received, processed and retransmitted with a timing control down to nanoseconds.